

第3章 SQL语言



沈明玉

合肥工业大学

Hefei University of Technology 计算机与信息学院



第3章 SQL语言

主要内容:

- 3.1 SQL概述
- 3.2 数据定义
- 3.3 数据查询
- 3.4 数据更新
- 3.5 视图

合肥工业大学

Hefei University of Technology 计算机与信息学院



第3章 SQL语言

3.1 SQL概述

3.1.1 SQL的产生与发展

- IBM, Sequel语言, 70年代, System R 的一部分; 后改名为 SQL;
- ANSI 和 ISO 发布的 SQL标准:
 - SQL-86、SQL-89、SQL-92、SQL:1999、SQL:2003、SQL:2008、SQL:2011
- SQL是关系数据库的标准语言, 是一个通用的、功能极强的关系数据库语言。

合肥工业大学

Hefei University of Technology 计算机与信息学院



3.1 SQL概述

3.1.2 SQL语言的特点

- 综合统一: SQL集数据定义语言 (DDL), 数据操纵语言 (DML), 数据控制语言 (DCL) 功能于一体。
- 可以独立完成数据库生命周期中的全部活动:
 - 定义关系模式, 插入数据, 建立数据库;
 - 对数据库中的数据进行查询和更新;
 - 数据库重构和维护
 - 数据库安全性、完整性控制等
- 非过程化语言、面向集合的操作方式。
- 提供交互式 and 嵌入式两种使用方法。
- 语言简洁、易学易用。

SQL 语言的动词

SQL 功能	动词
数据查询	SELECT
数据定义	CREATE, DROP, ALTER
数据操纵	INSERT, UPDATE, DELETE
数据控制	GRANT, REVOKE

合肥工业大学

Hefei University of Technology 计算机与信息学院



第3章 SQL语言

3.2 数据定义

3.2.1 SQL可定义的数据对象

- 模式 (架构) 的定义
- 基本表的定义
- 索引的定义
- 视图的定义
- 其它 (用户、触发器、过程与函数等)

合肥工业大学

Hefei University of Technology 计算机与信息学院



3.2 数据定义

3.2.2 架构 (Schema) 的定义与删除

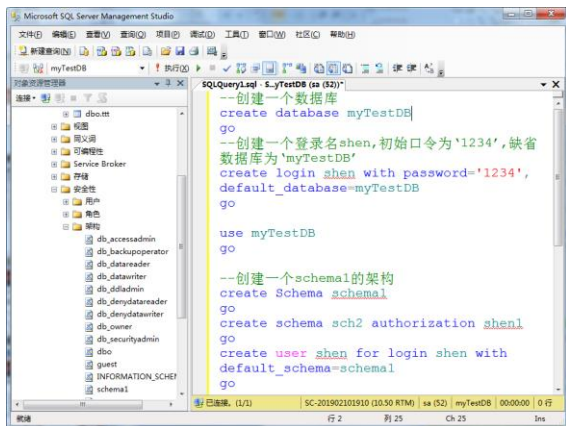
- 数据库架构是一个独立于数据库用户的非重复命名空间, 可将架构视为对象的容器。在这个空间中包含数据库的各种对象, 例如基本表、视图、索引等;



合肥工业大学

Hefei University of Technology 计算机与信息学院





3.2 数据定义

■ 删除架构

DROP SCHEMA <架构名> [CASCADE|RESTRICT]

✓ CASCADE(级联)

删除架构的同时把该架构中的所有数据库对象全部删除。

✓ RESTRICT(限制)

如果该架构中包含数据库对象（如表、视图等），则拒绝该删除语句的执行。

[例4] DROP SCHEMA ZHANG CASCADE;

合肥工业大学 Hefei University of Technology 计算机与信息学院

3.2 数据定义

3.2.3 基本表的创建、删除与修改

■ 创建基本表

CREATE TABLE <表名> (<列名> <数据类型> [<列级完整性约束条件>]

[, <列名> <数据类型> [<列级完整性约束条件>]] ...

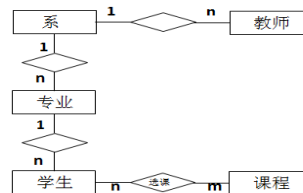
[, <表级完整性约束条件>]] ;

✓ 如果完整性约束条件涉及到该表的多个属性列，则必须定义在表级上，否则既可以定义在列级也可以定义在表级。

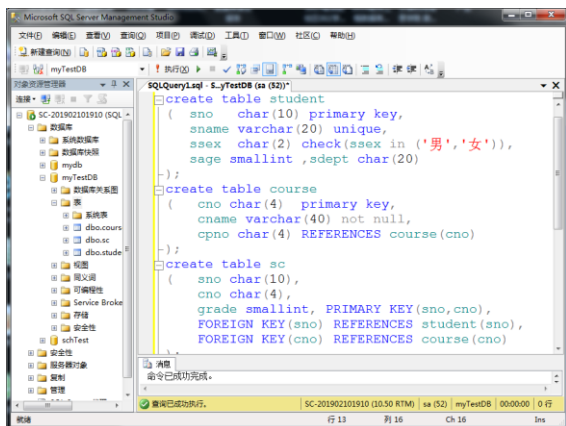
合肥工业大学 Hefei University of Technology 计算机与信息学院

3.2 数据定义

■ 简单的教学管理实体联系图



合肥工业大学 Hefei University of Technology 计算机与信息学院



3.2 数据定义

■ 修改基本表

ALTER TABLE <表名>

[ADD [column] <新列名> <数据类型> [完整性约束]]

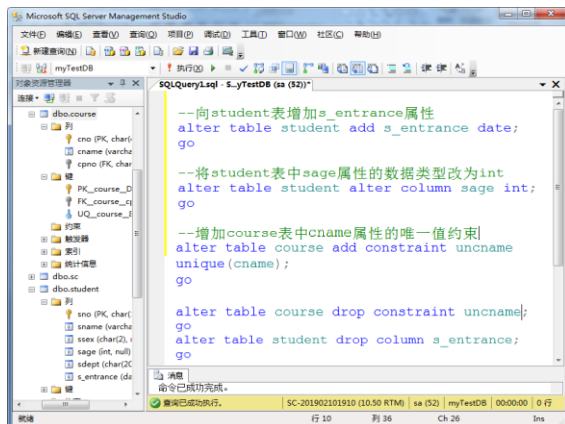
[ADD <表级完整性约束>]

[DROP [column] <列名> [CASCADE | RESTRICT]]

[DROP CONSTRAINT <完整性约束名> [CASCADE | RESTRICT]]

[ALTER COLUMN <列名> <数据类型>] ;

合肥工业大学 Hefei University of Technology 计算机与信息学院



3.2 数据定义

■ 删除基本表

`DROP TABLE <表名> [RESTRICT| CASCADE];`

☞ **RESTRICT**: 删除表是有限制的。

✓ 欲删除的基本表不能被其他表的约束所引用

✓ 如果存在依赖该表的对象, 则此表不能被删除

☞ **CASCADE**: 删除该表没有限制。

✓ 在删除基本表的同时, 相关的依赖对象一起删除

`DROP TABLE Student CASCADE;`

`DROP TABLE Student RESTRICT; // 需参照规定!`

Hefei University of Technology 计算机与信息学院

3.2 数据定义

3.2.4 索引的建立与删除

☞ 建立索引的目的: 加快查询速度

☞ 索引的种类: 顺序索引、B+树索引、哈希索引、位图索引等。

☞ 谁可以建立索引

- ✓ DBA 或 表的属主 (即建立表的人)
- ✓ DBMS 一般会 自动建立 以下列上的索引
 - PRIMARY KEY
 - UNIQUE

☞ 谁维护索引? DBMS 自动完成

☞ 使用索引

DBMS 自动选择是否使用索引以及使用哪些索引

Hefei University of Technology 计算机与信息学院

3.2 数据定义

■ 建立索引

`CREATE [UNIQUE] [CLUSTER] INDEX <索引名>
ON <表名>(<列名>[<次序>][<列名>[<次序>]]...);`

注意: 一个基本表上最多只能建立一个聚集索引。

```
create clustered index sname on student(sname);
go
```

消息 1902, 级别 16, 状态 3, 第 2 行
无法对表 'student' 创建多个聚集索引。请在创建新聚集索引前删除现有的聚集索引
'PK_student_DDDF64461BFDC07'。

Hefei University of Technology 计算机与信息学院

3.2 数据定义

```

CREATE UNIQUE INDEX Stusno ON Student(Sno);

CREATE UNIQUE INDEX Coucno ON Course(Cno);

CREATE UNIQUE INDEX SCno ON SC(Sno ASC,Cno DESC);

```



3.2 数据定义

■ 删除索引

`DROP INDEX <索引名> ON <表名>;`

删除索引时, 系统会从数据字典中删去有关该索引的描述。

```
drop index scno on sc
go
```



Hefei University of Technology 计算机与信息学院

第3章 SQL语言

3.3 数据查询

语句格式:

选择(σ) **投影(π)** **笛卡尔积(\times)**

```
SELECT [ALL|DISTINCT] <目标列表达式>[, <目标列表达式>]...
FROM <表名或视图名>[, <表名或视图名>]...
WHERE <条件表达式>
[GROUP BY <列名1> [HAVING <条件表达式>]]
[ORDER BY <列名2> [ASC|DESC]];
```

<目标列表达式>可以为: 算术表达式、字符串常量、函数、列别名

基本查询结构

Hefei University of Technology 计算机与信息学院

3.3 数据查询

查询条件

查询条件	谓词
比较	=, >, <, >=, <=, !=, <>, !=, < >, < > NOT+上述比较运算符
确定范围	BETWEEN AND, NOT BETWEEN AND
确定集合	IN, NOT IN
字符匹配	LIKE, NOT LIKE
空值	IS NULL, IS NOT NULL
多重条件(逻辑运算)	AND, OR, NOT

Hefei University of Technology 计算机与信息学院

3.3 数据查询

3.3.1 单表查询

SQLQuery1.sql - SC_910.mydb (sa (32))
-- 查询全体学生的学号、姓名和专业号
select sno 学号, sname 姓名, spno 专业号 from t_st
go

Select 子句的变化

3.3 数据查询

SQLQuery1.sql - SC_910.mydb (sa (32))
-- 查询所有选修了课程的学生学号, 取消重复行
select distinct sno from sc
go

Select 子句的变化

3.3 数据查询

3.3 数据查询

条件查询 (where 子句的使用与变化)

SQLQuery3.sql - SC_910.mydb (sa (32))
-- 查询计算机专业(01)全体学生信息
select * from t_st where spno='01'
go

sno	sname	sex	sage	spno
1	20160001	宋江	20	01
2	20160002	卢俊义	21	01
3	20160003	吴用	19	01
4	20160004	公孙胜	18	01
5	20160005	关胜	19	01
6	20160006	林冲	20	01
7	20160007	秦明	19	01

3.3 数据查询

SQLQuery3.sql - SC_910.mysdb (sa (52))*

```
-- 查询软件工程 (03) 专业全体男生的信息
select * from t_st where spno='03' and
ssex='男'
go
```

Where 子句的使用与变化

sno	sname	ssex	sage	spno	
1	20160070	孟康	男	23	03
2	20160071	倬健	男	21	03
3	20160072	陈达	男	20	03
4	20160073	杨春	男	20	03
5	20160074	郑天寿	男	21	03
6	20160075	陶宗旺	男	23	03
7	20160076	宋清	男	21	03

3.3 数据查询

SQLQuery3.sql - SC_910.mysdb (sa (52))*

```
select * from t_st where sage between 20
and 22
go

select * from t_st where sage in (20,22)
go
```

条件中使用谓词(not) between...and

条件中使用谓词 in (not in)

sno	sname	ssex	sage	spno	
1	20160001	宋江	男	20	01
2	20160006	林中	男	20	01
3	20160020	戴宗	男	20	01
4	20160026	李俊	男	20	01
5	20160030	张顺	男	20	01
6	20160037	朱武	男	20	01
7	20160039	孙立	男	22	01

3.3 数据查询

字符串匹配:

[NOT] LIKE '<匹配串>' [ESCAPE '<换码字符>']

SQLQuery1.sql - SC_910.mysdb (sa (52))*

```
select * from t_st where sname like '宋_'
```

通配符_的使用

sno	sname	ssex	sage	spno	
1	20160001	宋江	男	20	01
2	20160076	宋清	男	21	03
3	20160082	宋万	男	19	03

3.3 数据查询

SQLQuery1.sql - SC_910.mysdb (sa (52))*

```
select * from t_st where sname like '%胜%'
```

通配符%的使用

sno	sname	ssex	sage	spno	
1	20160004	公孙胜	男	18	01
2	20160005	关胜	男	19	01
3	20160106	白胜	男	21	04

合肥工业大学 计算机与信息学院

3.3 数据查询

涉及空值的查询:

IS NULL 或 IS NOT NULL

注: "is" 不能用 "=" 代替

SQLQuery1.sql - SC_910.mysdb (sa (52))*

```
select * from sc where cno='c104' and grade
is not null
go
```

条件中空值的判断

sno	cno	grade	
1	20160001	c104	90
2	20160002	c104	88
3	20160003	c104	95
4	20160004	c104	89

3.3 数据查询

对结果进行排序的查询 (ORDER BY 属性名 desc[,...])

- ✓ 可以按一个或多个属性列排序: 升序(缺省): ASC; 降序: DESC;
- ✓ 当排序列含空值时:
 - ◆ ASC: 排序列为空值的元组最后显示;
 - ◆ DESC: 排序列为空值的元组最先显示。

SQLQuery1.sql - SC_administrator (323)*

```
--结果排序: 专业号升序, 年龄降序
select * from t_st order by spno,sage desc
go
```

sno	sname	ssex	sage	spno	
1	20160039	孙立	男	22	01
2	20160051	杨林	男	22	01
3	20160052	凌振	男	21	01
4	20160047	裴宣	男	21	01
5	20160040	宣赞	男	21	01
6	20160038	黄信	男	21	01

3.3 数据查询

基于统计的查询 (聚合函数)

- ✓ 计数
 - COUNT ([DISTINCT | ALL] *)
 - COUNT ([DISTINCT | ALL] <列名>)
- ✓ 计算总和
 - SUM ([DISTINCT | ALL] <列名>)
- ✓ 计算平均值
 - AVG ([DISTINCT | ALL] <列名>)
- ✓ 最大最小值
 - MAX ([DISTINCT | ALL] <列名>)
 - MIN ([DISTINCT | ALL] <列名>)

Hefei University of Technology 计算机与信息学院

3.3 数据查询

Count计算元组的个数，
注意参数的使用！

```
SQLQuery1.sql - SC_administrator (52)*
select COUNT(*) from t_st
go

select COUNT(spno) from t_st
go

select COUNT(distinct spno) from t_st
go
```

3.3 数据查询

```
select sum(grade) from sc
where cno='c104'
go

select avg(grade) from sc
where cno='c104'
go
```

Sum属性值求和，
忽略空值

Sum属性值求平均
值，忽略空值

3.3 数据查询

```
select max(grade) from sc
where cno='c104'
go

select min(grade) from sc
where cno='c104'
go
```

Max找出属性值的
最大值，忽略
空值

Max找出属性值的
最小值，忽略
空值

3.3 数据查询

基于分组的统计查询 (GROUP BY子句)

- ✓ 细化聚合函数的作用对象；
- ✓ 作用对象是查询的中间结果表；
- ✓ 按指定的一列或多列值分组，值相等的为一组。

```
SELECT Cno,COUNT(Sno) FROM SC GROUP BY Cno
go
```

Cno	(无列名)
1	c101 1
2	c102 1
3	c103 1
4	c104 51

3.3 数据查询

Sname未出现在
Group子句中，错误！

```
select sname,COUNT(*) from t_st
group by spno
go

select spno,COUNT(*) from t_st
group by spno
go
```

消息 8120，级别 16，状态 1，第 2 行
选择列表中的列 't_st.sname' 无效，因为该列没有包含在聚合函数或 GROUP BY 子句中。

Select子句的正
确使用方法

spno	(无列名)
1	01 52
2	02 17
3	03 20
4	04 19

3.3 数据查询

3.3.2 多表连接查询

◆ 连接操作的几种执行方式

- ✓ 嵌套循环法（最耗时，无需条件）
- ✓ 排序合并法（最快，条件苛刻，理想化）
- ✓ 索引连接（最常用，效率优于嵌套循环，需条件）

Hefei University of Technology 计算机与信息学院

3.3 数据查询

■ 笛卡尔积（无连接条件，R1 cross join R2）

-- 查询已经选课的学生名单

```
select distinct t_st.sno, sname from t_st
cross join sc
go
```

sno	sname
20160001	宋江
20160002	卢俊义
20160003	吴用
20160004	公孙胜
20160005	关胜

-- 查询已经选课的学生名单

```
select distinct t_st.sno, sname from t_st, sc
go
```

sno	sname
20160001	宋江
20160002	卢俊义
20160003	吴用
20160004	公孙胜
20160005	关胜

3.3 数据查询

■ 等值连接

-- 查询学生的所选课程编号

```
select t_st.sno, sname, cno from t_st join sc
on (t_st.sno=sc.sno)
go
```

sno	sname	cno
20160001	宋江	c101
20160001	宋江	c102
20160001	宋江	c103
20160001	宋江	c104
20160001	宋江	c201

```
select t_st.sno, sname, cno from t_st, sc
where t_st.sno=sc.sno
go
```

sno	sname	cno
20160001	宋江	c101
20160001	宋江	c102
20160001	宋江	c103
20160001	宋江	c104
20160001	宋江	c201

3.3 数据查询

■ 自然连接

-- 查询选课学生的信息

```
select * from t_st natural join sc
go
```

消息 102，级别 16，状态 1，第 2 行
“sc”，附近有语法错误。

-- 查询选课学生的信息

```
select * from t_st join sc on (t_st.sno=sc.sno)
go
```

sno	sname	sex	age	spno	sno	cno	grade
20160001	宋江	男	20	01	20160001	c101	NULL
20160001	宋江	男	20	01	20160001	c102	NULL
20160001	宋江	男	20	01	20160001	c103	NULL
20160001	宋江	男	20	01	20160001	c104	90

3.3 数据查询

■ 自身连接

-- 查询同时选修了 'c104' 和 'c210' 课程的学生学号

```
select sc1.sno from sc sc1, sc sc2
where sc1.sno=sc2.sno and sc1.cno='c104'
and sc2.cno='c210'
go
```

sno
20160001
20160002
20160003
20160004
20160005
20160006
20160007
20160008

3.3 数据查询

■ 外连接 (Outer join)

左外连接: left outer join

右外连接: right outer join

全外连接: full outer join

-- 左外连接，注意条件表达式！

```
select t_st.sno, sname, cno from t_st left
join sc on (t_st.sno=sc.sno and cno='c210')
go
```

sno	sname	cno
20160007	汤隆	c210
20160008	杜兴	c210
20160009	郭润	c210
20160009	郭润	NULL
20160091	朱贵	NULL
20160092	朱富	NULL
20160093	施恩	NULL

3.3 数据查询

■ 多表连接 (Outer join)

```
-- 多表连接
select t_st.sno,sname,t_d.dno,dname
from t_st,t_sp,t_d
where t_st.spno=t_sp.spno
and t_sp.dno=t_d.dno
go
```

sno	sname	dno	dname
1	20160001	宋江	0001 计算机
2	20160002	卢俊义	0001 计算机
3	20160003	吴用	0001 计算机
4	20160004	公孙胜	0001 计算机
5	20160005	关胜	0001 计算机
6	20160006	林冲	0001 计算机

3.3 数据查询

4.3.3 嵌套查询

- ✓ 一个SELECT-FROM-WHERE语句称为一个**查询块**。
- ✓ 将一个查询块嵌套在另一个查询块的WHERE子句或HAVING短语的条件中。
- ✓ 子查询中不能使用ORDER BY子句。
- ✓ 相关子查询与不相关子查询的区别与处理。

```
-- 嵌套查询
select sno,sname from t_st
where sno in (select sno from sc
where cno='c104')
go
```

sno	sname
1	20160001 宋江
2	20160002 卢俊义
3	20160003 吴用

3.3 数据查询

■ 带有IN谓词的子查询

```
-- 嵌套查询:带有IN谓词的子查询
select * from t_st
where spno in
(select spno from t_st where sname='李逵')
go
```

sno	sname	ssex	sage	spno
1	20160001	宋江	男	20 01
2	20160002	卢俊义	男	21 01
3	20160003	吴用	男	19 01
4	20160004	公孙胜	男	18 01
5	20160005	关胜	男	19 01
6	20160006	林冲	男	20 01
7	20160007	秦明	男	19 01
8	20160008	呼延灼	男	18 01

3.3 数据查询

■ 带有比较运算符的子查询

- ◆ 当能确切知道内层查询返回**单值**时,可用比较运算符(>, <, =, >=, <=, !=或<>)。

```
-- 嵌套查询:带有比较运算符的子查询
select * from t_st
where spno =
(select spno from t_st where sname='李逵')
go
```

- ◆ 与ANY或ALL谓词配合使用。
(参见教材 P107)

合肥工业大学 计算机与信息学院

3.3 数据查询

[例] 查询其他专业中比软件工程专业任意一个(其中某一个)学生年龄小的学生。

```
-- 嵌套查询:带有ANY谓词的子查询
SELECT * FROM t_st
WHERE sage <ANY (SELECT sage FROM t_st
WHERE spno='03')
AND spno <> '03' ;
```

sno	sname	ssex	sage	spno
1	20160001	宋江	男	20 01
2	20160002	卢俊义	男	21 01
3	20160003	吴用	男	19 01
4	20160004	公孙胜	男	18 01
5	20160005	关胜	男	19 01
6	20160006	林冲	男	20 01
7	20160007	秦明	男	19 01
8	20160008	呼延灼	男	18 01
9	20160009	花荣	男	16 01
10	20160010	柴进	男	21 01

3.3 数据查询

● ANY和ALL谓词有时可以用聚集函数实现

	=	<>或!=	<	<=	>	>=
ANY	IN	--	<MAX	<=MAX	>MIN	>=MIN
ALL	--	NOT IN	<MIN	<=MIN	>MAX	>=MAX

```
-- 嵌套查询:用聚集函数等价实现
SELECT * FROM t_st
WHERE sage < (SELECT max(sage) FROM t_st
WHERE spno='03')
AND spno <> '03' ;
```

- ✓ 用聚集函数实现子查询通常比直接用ANY或ALL查询效率高,因为前者通常能够减少比较次数。

合肥工业大学 计算机与信息学院

3.3 数据查询

■ 带有EXISTS谓词的子查询

1. EXISTS谓词

格式: `Select sno,sname From student Where Exists (Select * from sc Where sc.sno=student.sno And cno='c104');`

- 带有EXISTS谓词的子查询不返回任何数据
 - 若内层查询结果非空, 则为真
 - 若内层查询结果为空, 则为假
- 由EXISTS引出的子查询, 其目标表达式用*, 给出列名无实际意义。

2. NOT EXISTS谓词

合肥工业大学 计算机与信息学院

3.3 数据查询

```
-- 查询所有选修了课程号为 'c210' 的学生姓名
-- 使用 Exists 谓词实现 ( 相关子查询 )
select sname from t_st where exists
(select * from sc where sc.sno=t_st.sno
and cno='c210');

-- 使用 in 谓词等价实现 ( 不相关子查询 )
select sname from t_st where sno in
(select sno from sc where cno='c210');
```

结果	消息
sname	
1 宋江	
2 卢俊义	
3 吴用	
4 公孙胜	
5 关胜	
6 林冲	

3.3 数据查询

■ 查询选修了全部课程的学生姓名

查找这样的学生: 对于课程表中的每一门课程, 该生都选了 (即: 均可在SC表中找到该学生选修这门课的元组), 也可表示成: 不存在这样的课程, 在选课表SC中不存在该学生对该课程的选课元组)。

```
-- 查询选修了全部课程号的学生姓名
select sname from t_st where not exists
(select * from t_c where not exists
(select * from sc where sc.sno=t_st.sno
and sc.cno=t_c.cno));
```

结果	消息
sname	
1 宋江	

3.3 数据查询

■ 查询至少选修了学生20160002所选全部课程的学生姓名

查找这样的学生: 对于课程表中的每一门课程, 如果该课程被20160002学生选修了 (SC中存在), 一定可以在SC中找到该生这门课的选课元组。

```
-- 查询至少选修了学生20160002选修的全部课程的学生姓名。
select sname from t_st
where not exists (select * from sc sc1
where sc1.sno='20160002' and not exists
(select * from sc sc2
where sc2.sno=t_st.sno and
sc2.cno=sc1.cno));
```

结果	消息
sname	
1 宋江	
2 卢俊义	
3 吴用	
4 公孙胜	
5 关胜	
6 林冲	

3.3 数据查询

4.3.4 集合查询

- ✓ 并操作UNION
- ✓ 交操作INTERSECT
- ✓ 差操作EXCEPT

注: 参加集合操作的各查询结果的列数必须相同; 对应项的数据类型也必须相同。

合肥工业大学 计算机与信息学院

3.3 数据查询

```
-- 查询非计算机系或未选修了课程的学生学号与姓名
select sno,sname from t_st
where spno in (select spno from t_sp
where dno in (select dno from t_d
where dname<>'计算机'))
union
select sno,sname from t_st where sno not in
(select t_st.sno from t_st,sc where t_st.sno=sc.sno);
```

结果	消息
sno	sname
1 20160070	孟康
2 20160071	侯健
3 20160072	陈达
4 20160073	杨春
5 20160074	郑天寿
6 20160075	陶宗旺

3.3 数据查询

--查询同时选修了'c210'和'c211'课程的学生学号与姓名

```
select sno,sname from t_st
where sno in (
    (select sno from sc where cno='c210')
    intersect
    (select sno from sc where cno='c211'));
```

结果 消息

sno	sname
1 20160001	宋江

3.3 数据查询

--查询没有选修'c211'课程的学生学号

```
select sno from t_st
except
select sno from sc where cno='c211';
```

结果 消息

sno
1 20160002
2 20160003
3 20160004
4 20160005
5 20160006
6 20160007
7 20160008
8 20160009
9 20160010
10 20160011
11 20160012
12 20160013
13 20160014

3.3 数据查询

3.3.5 基于派生表的查询

✓ 查询子句出现在from子句中，查询结果作为操作对象

-- 查询计算机系且选修了'c217'课程的学号与姓名

```
select sc.sno,sname from sc,
(select sno,sname from t_st
where spno in (select spno from t_sp
where dno=(select dno from t_d
where dname='计算机')) as jsj
where sc.sno=jsj.sno and cno='c217');
```

结果 消息

sno	sname
1 20160001	宋江
2 20160002	凌康
3 20160003	蒋敬
4 20160004	吕方
5 20160005	郭盛

第3章 SQL语言

3.4 数据更新

- 1、插入数据 （INSERT 语句）
- 2、修改数据 （UPDATE 语句）
- 3、删除数据 （DELETE 语句）

注意：数据更新操作中的完整性控制！

合肥工业大学 计算机与信息学院



3.4 数据更新

1、插入数据

(1) 插入元组

```
INSERT INTO <表名> [(<属性列1>[, <属性列2>...])]
VALUES (<常量1>[, <常量2> ...]);
```

(2) 插入子查询结果

```
INSERT INTO <表名> [(<属性列1>[, <属性列2>...])] 子查询;
```

合肥工业大学 计算机与信息学院



3.4 数据更新

-- 数据插入

```
insert into t_st
values ('20160109','张三','男',21,'01'),
('20160110','李四','男',22,'02');

create table sp_age
(sjno char(2), avg_age numeric(4,1));

insert into sp_age(sjno,avg_age)
select sjno,avg(sage) from t_st
group by sjno;

select * from sp_age;
```

结果 消息

sno	avg_age
1 01	19.0
2 02	20.0
3 03	20.0
4 04	19.0

3.4 数据更新

2、修改数据

UPDATE <表名> SET <列名>=<表达式>[, <列名>=<表达式>]...

[WHERE <条件>];

✓ 功能：修改指定表中满足WHERE子句条件的元组。

✓ 方式：

- (1) 修改某一个元组的值
- (2) 修改多个元组的值
- (3) 带子查询的修改

合肥工业大学

计算机与信息学院



3.4 数据更新

```
-- 修改高俅的手机号
update t_t set tphone='18905510123'
where tname='高俅';

-- 将'03'专业学生'c210'课程的成绩修改为80分
update sc set Grade=80
where cno='c210' and '03'=(
select spno from t_st
where t_st.sno = sc.sno);
select * from sc where cno='c210';
```

sno	cno	grade
69	20160069	c210 NULL
70	20160070	c210 80.0
71	20160071	c210 80.0
72	20160072	c210 80.0
73	20160073	c210 80.0

3.4 数据更新

3、删除数据

DELETE FROM <表名> [WHERE <条件>];

• 功能

- 删除指定表中满足WHERE子句条件的元组。

• WHERE子句

- 指定要删除的元组；
- 缺省表示要删除表中的所有元组，表的定义仍在字典中。

合肥工业大学

计算机与信息学院



3.4 数据更新

```
-- 删除学生张三的信息
delete from t_st where sname='张三';

-- 将'03'专业学生'c210'课程的成绩修改为80分
delete from sc
where cno='c210' and '03'=(
select spno from t_st
where t_st.sno = sc.sno);
```

第3章 SQL语言

3.5 视图

■ 理解视图

- ✓ 虚表，是从一个或几个基本表（或视图）导出的表
- ✓ 只存放视图的定义，不存放视图对应的数据
- ✓ 基表中的数据发生变化，从视图中查询出的数据也随之改变
- ✓ 基于视图的操作：

查询、删除、受限更新、定义基于该视图的新视图

合肥工业大学

计算机与信息学院



3.5 视图

3.5.1 定义视图

1. 创建视图

语句格式：

CREATE VIEW <视图名> [(<列名> [, <列名>]...)]

AS <子查询> [WITH CHECK OPTION];

- ✓ 视图定义中的属性列表；
- ✓ 视图定义中子查询的ORDER BY和DISTINCT；
- ✓ 视图定义后存入数据字典，并不执行其中的SELECT语句；
- ✓ WITH CHECK OPTION 选项。

合肥工业大学

计算机与信息学院



3.5 视图

```
SQLQuery2.sql - SC...dministrator (54) SQLQuery1.sql - SC...Administrator (52)
/* 定义视图计算机系的学生视图 */
Create View V_jsj AS
Select * from t_st
Where spno in (Select spno from t_sp
Where dno =(Select dno from t_d
Where dname='计算机'));
```

命令已成功完成。

3.5 视图

```
/* 视图定义中使用表达式 */
Create View V_Student_BYear As
Select Sno, Sname, Ssex,
year(getdate())-Sage byear
from t_st;
```

命令已成功完成。

```
SQLQuery2.sql - SC...dministrator (54) SQLQuery1.sql - SC...Administrator (52)
Select * from V_Student_BYear
```

Sno	Sname	Ssex	byear
20160001	宋江	男	2000
20160002	卢俊义	男	1999
20160003	吴用	男	2001
20160004	公孙胜	男	2002

3.5 视图

```
/* 定义使用聚集函数的视图 */
Create View V_SC_AVG_80 AS
Select t_st.Sno, Sname, AVG(Grade) cavg
from t_st, sc Where t_st.sno=sc.sno
group by t_st.Sno, Sname
Having AVG(Grade) >=80;
```

命令已成功完成。

```
Select * from V_SC_AVG_80;
```

Sno	Sname	cavg
20160001	宋江	90.000000
20160002	卢俊义	88.000000
20160003	吴用	95.000000
20160004	公孙胜	89.000000
20160005	关胜	89.000000
20160007	秦明	91.000000

3.5 视图

```
/* 定义一个行列子集视图 */
Create View V_SC_C208 AS
Select * from sc Where cno='c208';
```

命令已成功完成。

```
Select * from V_SC_C208;
```

sno	cno	grade
20160001	c208	NULL
20160002	c208	NULL
20160003	c208	NULL
20160004	c208	NULL
20160005	c208	NULL
20160006	c208	NULL
20160007	c208	NULL

3.5 视图

```
/* 基于其他视图定义新视图 */
Create View V_jsj_c208 AS
Select V_jsj.*, cno, grade from V_jsj,
V_SC_C208
Where V_jsj.sno=V_SC_C208.sno;
```

命令已成功完成。

3.5 视图

```
/* 定义一个学生、专业、系关系视图 */
Create View v_smd As
Select sno, sname, t_st.spno, spname, t_d.dno,
dname from t_st, t_sp, t_d
Where t_st.spno=t_sp.spno and
t_sp.dno=t_d.dno;
```

命令已成功完成。

```
select * from v_smd;
```

sno	sname	spno	spname	dno	dname
20160001	宋江	01	计算机科学与技术	0001	计算机
20160002	卢俊义	01	计算机科学与技术	0001	计算机
20160003	吴用	01	计算机科学与技术	0001	计算机
20160004	公孙胜	01	计算机科学与技术	0001	计算机
20160005	关胜	01	计算机科学与技术	0001	计算机
20160006	林冲	01	计算机科学与技术	0001	计算机
20160007	秦明	01	计算机科学与技术	0001	计算机
20160008	呼延灼	01	计算机科学与技术	0001	计算机

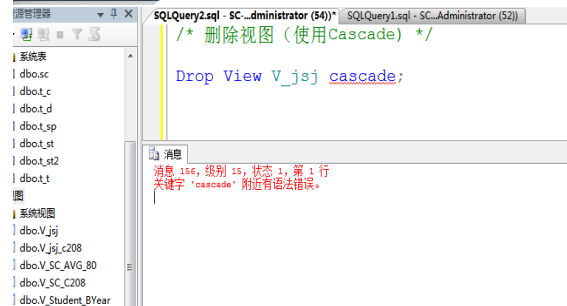
3.5 视图

2. 删除视图

DROP VIEW <视图名>[CASCADE];

- ✓ 该语句从数据字典中删除指定的视图定义。
- ✓ 如果该视图上还导出了其他视图，使用CASCADE级联删除语句，把该视图和由它导出的所有视图一起删除。
- ✓ 删除基表时，由该基表导出的所有视图定义都必须显式地使用DROP VIEW语句删除。

3.5 视图



3.5 视图

3.5.2 查询视图

- 用户角度：查询视图与查询基本表相同
- RDBMS实现视图查询的方法：

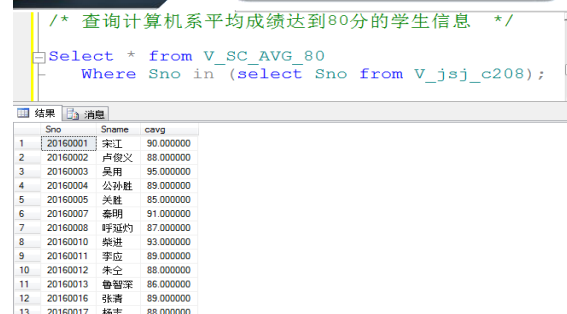
视图消解法：

- ✓ 进行有效性检查；
- ✓ 转换成等价的对基本表的查询；
- ✓ 执行修正后的查询。

实体化视图：

- ✓ 有效性检查；
- ✓ 执行视图定义，生成临时表；
- ✓ 查询视图转化为查询临时表；
- ✓ 查询完后删除实体化得视图。

3.5 视图



3.5 视图

● 视图消解法的局限

有些情况下，视图消解法不能生成正确查询。

Create View v_avg as select sno, avg(grade) cavg
from sc group by sno;

使用：

SELECT * FROM v_avg WHERE cavg>=90;

✗: SELECT Sno, AVG(Grade) FROM SC
WHERE AVG(Grade)>=90 GROUP BY Sno;

✓: SELECT Sno,AVG(Grade) FROM SC
GROUP BY Sno HAVING AVG(Grade)>=90;

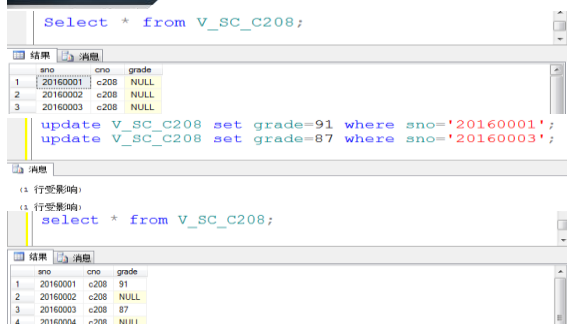
3.5 视图

3.5.3 更新视图

更新视图的限制：

- ✓ 一些视图是不可更新的，这些视图的更新无法转换成对相应基本表的更新。
- ✓ 允许对行列子集视图进行更新
- ✓ 对其他类型视图的更新不同系统有不同限制

3.5 视图



```

Select * from V_SC_C208;

```

ano	cno	grade
20160001	c208	NULL
20160002	c208	NULL
20160003	c208	NULL

```

update V_SC_C208 set grade=91 where sno='20160001';
update V_SC_C208 set grade=87 where sno='20160003';

```

```

select * from V_SC_C208;

```

ano	cno	grade
20160001	c208	91
20160002	c208	NULL
20160003	c208	87
20160004	c208	NULL

3.5 视图

3.5.4 视图的作用

1. 视图能够简化用户的操作;
2. 视图使用户能以多种角度看待同一数据;
3. 视图对重构数据库提供了一定程度的逻辑独立性;
4. 视图能够对机密数据提供安全保护;
5. 适当的利用视图可以更清晰的表达查询。

合肥工业大学 Hefei University of Technology 计算机与信息学院

第3章 SQL语言

■ 本章思考题:

如何理解数据库中的架构 (Schema) ?

■ 本章作业:

P130 习题3、习题5

合肥工业大学 Hefei University of Technology 计算机与信息学院